MOVEBIT
Securing the Move Ecosystem

KriyaDEX Smart Contract **Audit Report**

https://twitter.com/movebit_

contact@movebit.xyz

KriyaDEX Smart Contract Audit Report

MOVEBIT

1 Executive Summary

1.1 Project Information

Description	An AMM DEX on SUI
Type	DEX
Auditors	MoveBit
Timeline	Apr 27, 2023 – May 2, 2023
Languages	Move
Platform	Sui
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/efficacy-finance/sui-spot-amm-modules
Commits	a8371b4138eda4ba0170ad0bc36a42dc9ffbde38 ed89807de337c060e5ba69e74cff23e81552e28c 741ca21ac23c50e0fe7a280f0a8e8b190b16a03b 260e7d0d02a4c616bca67beb890a6f13c861ab4e

1.2 Files in Scope

The following are the SHA1 hashes of the last reviewed files.

ID	Files	SHA-1 Hash
MATH	sources/safe_math.move	ba32f8b0c05df857719b91502ab8af4d0dd725c0
UTILS	sources/utils.move	b658bad29e1e6a1e15531b42fea477ff723422d7
SPOT	sources/spot_dex.move	e922c9159f82145b20ecfa77a166eb8b1c33f76f

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	11	11	
Informational			
Minor	6	6	
Medium	2	2	
Major	3	3	
Critical			

1.4 MoveBit Audit BreakDown

MoveBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control

- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" and "**Formal Verification**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section **1.2**.

(3) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by **Kriya** to identify any potential issues and vulnerabilities in the source code of the **KriyaDEX** smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified **11** issues of varying severity, listed below.

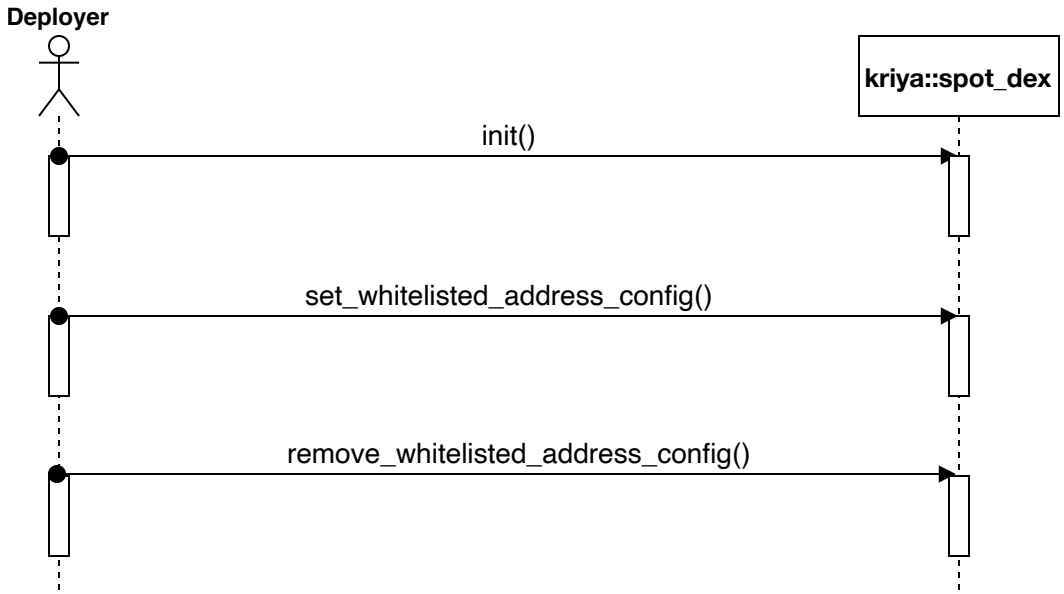
ID	Title	Severity	Status
SPOT-01	Receive Return Values in the Incorrect Order	Major	Fixed
SPOT-02	Calculation Formula Error When Adding Liquidity	Major	Fixed
SPOT-03	Validating Errors When Adding to the Whitelist	Major	Fixed
SPOT-04	Lack of Minimum Liquidity	Medium	Fixed
SPOT-05	Missing K Value Verification	Medium	Fixed
SPOT-06	Unused Variable	Minor	Fixed
SPOT-07	Redundant `assert` Statements	Minor	Fixed
SPOT-08	Sensitive Operations Require Adding an Event	Minor	Fixed
SPOT-09	Parameter Limit	Minor	Fixed
SPOT-10	The Condition of the Assert is Incorrect	Minor	Fixed
SPOT-11	Incorrect Order of Variables	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the `KriyaDEX` Smart Contract :

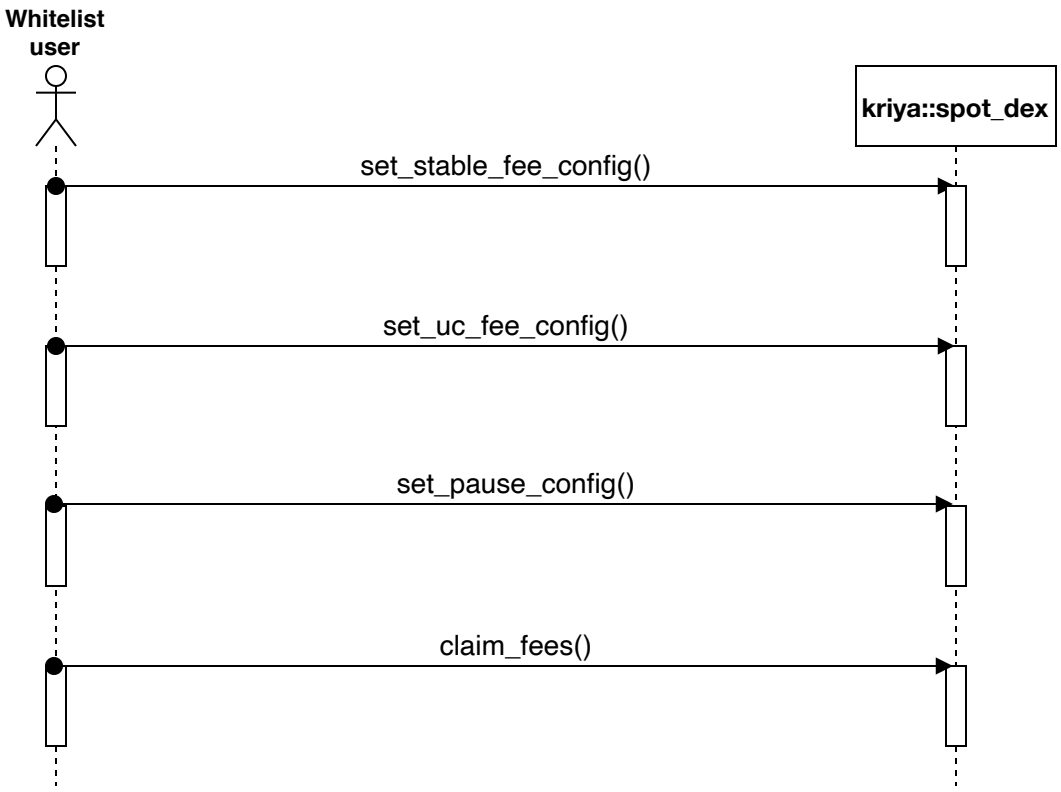
Deployer

- Deployer can create `ProtocolConfigs` in deployment through `init()`.
- Deployer can add whitelist addresses `through set_whitelisted_address_config()`.
- Deployer can delete whitelisted addresses through `remove_whitelisted_address_config()`.



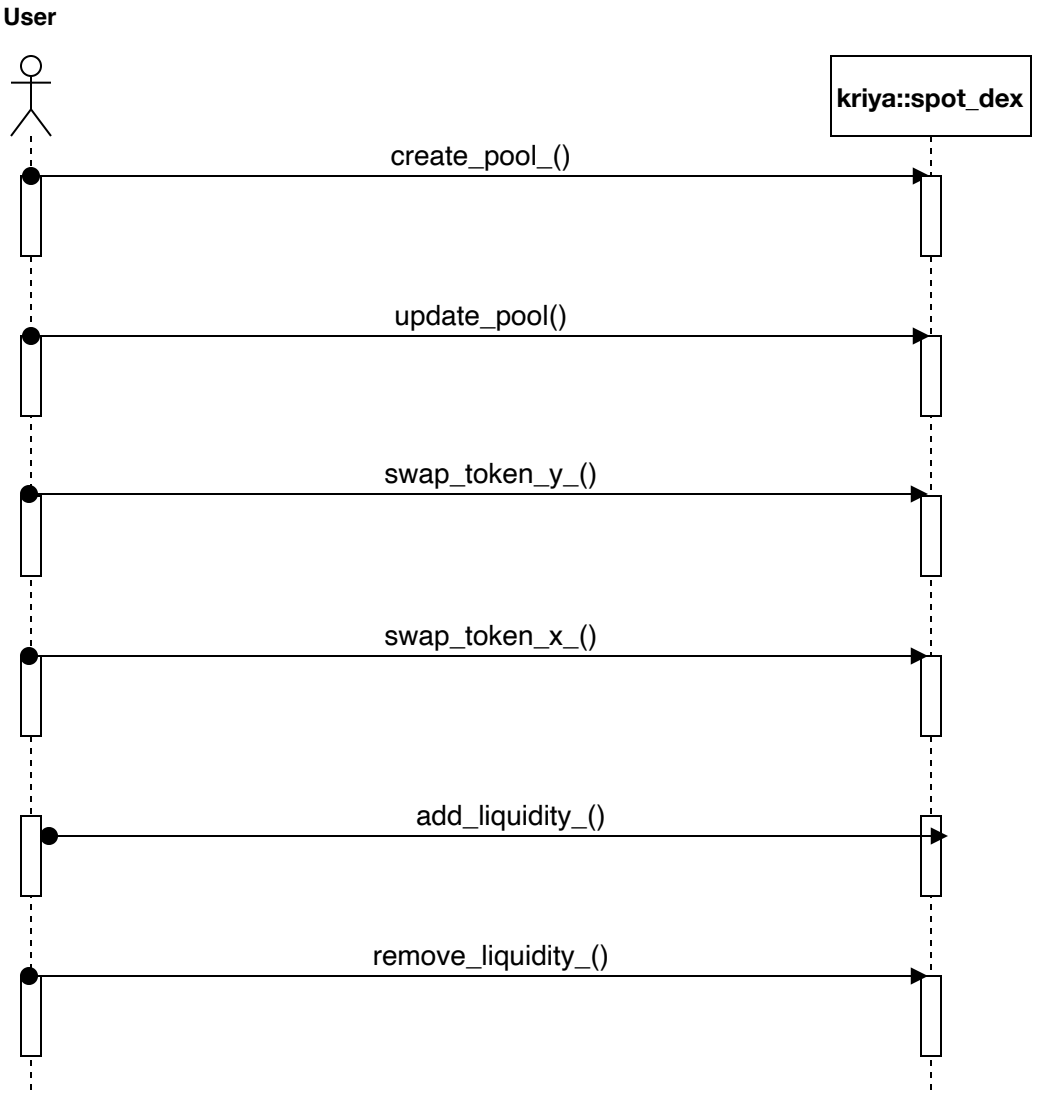
Whitelist user

- Whitelist users can set the fee rate through `set_stable_fee_config()` and `set_uc_fee_config()`.
- Whitelist users can withdraw fees through `claim_fees()`.
- Whitelist users can set pause configuration in `ProtocolConfigs` through `set_pause_config()`.



User

- User can create pools through `create_pool_()`.
- User can update the pool through `update_pool()`.
- User can swap tokens through `swap_token_y_()` and `swap_token_x_()`.
- User can add liquidity through `add_liquidity_()`.
- User can remove liquidity through `remove_liquidity_()`.



4 Findings

SPOT-01 Receive Return Values in the Incorrect Order

Severity: Major

Status: Fixed

Code Location: sources/spot_dex.move, L284,L341,L691,L724.

Descriptions: The incorrect order of the return values of the `get_reserves()` function will result in incorrect calculation of amounts during swapping and adding liquidity.

Suggestion: Adjust the order of the return values received.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-02 Calculation Formula Error When Adding Liquidity

Severity: Major

Status: Fixed
Code Location: sources/spot_dex.move, L689.
Descriptions: The formula for obtaining the value of the other token quantity through one token quantity in the function `get_amount_for_add_liquidity()` is incorrect, resulting in incorrect calculation results and affecting the assets invested by users when adding liquidity.
Suggestion: Adjust the formula to ensure that the result of adding liquidity is correct.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-03 Validating Errors When Adding to the Whitelist

Severity: Major

Status: Fixed
Code Location: sources/spot_dex.move, L447.
Descriptions: The condition in the `assert` statement is reversed when adding to the whitelist in the function `set_whitelisted_address_config()`, which prevents the addition of a new address to the whitelist.
Suggestion: Modify the condition within the `assert` statement.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-04 Lack of Minimum Liquidity

Severity: Medium

Status: Fixed
Code Location: sources/spot_dex.move, L510.
Descriptions: In the function `add_liquidity()` no minimum liquidity was added when adding liquidity for the first time.
Suggestion: When adding liquidity for the first time, add the minimum liquidity and lock it in the pool.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-05 Missing K Value Verification

Severity: Medium

Status: Fixed
Code Location: sources/spot_dex.move.
Descriptions: After executing the swap, the value of K should be validated to be greater than or equal to the previous value of K.
Suggestion: Add an `assert` verification to ensure that the new value of K is greater than or equal to the previous value of K.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-06 Unused Variable

Severity: Minor

Status: Fixed
Code Location: sources/spot_dex.move, L146.
Descriptions: The parameter `ctx` of the function `update_pool()` is not being utilized, which may result in warnings or unnecessary memory consumption.
Suggestion: It is suggested to rename `ctx` to `_ctx` to indicate that it is an unused parameter and to avoid any potential issues.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-07 Redundant `assert` Statements

Severity: Minor

Status: Fixed
Code Location: sources/spot_dex.move, L498.
Descriptions: The `assert` validation always passes when adding liquidity because its condition is the same as the previous if statement.
Suggestion: Delete this `assert`.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-08 Sensitive Operations Require Adding an Event

Severity: Minor

Status: Fixed

Code Location: sources/spot_dex.move.

Descriptions: Sensitive operations require adding an event.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-09 Parameter Limit

Severity: Minor

Status: Fixed

Code Location: sources/spot_dex.move, L171,L196.

Descriptions: The parameters `scaleX` and `scaleY` of the function `create_pool` may pose a risk if they are freely inputted by the user.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-10 The Condition of the Assert is Incorrect

Severity: Minor

Status: Fixed

Code Location: sources/spot_dex.move, L257,L315,L466.

Descriptions: In the function `swap_token_y()`, the validation should be for `amount > 0` instead of `token_y_balance > 0`. The same issue also exists in lines 315 and 466.

Suggestion: Modify the condition in the `assert` statement.

Resolution: The client has followed our suggestion and fixed the issue.

SPOT-11 Incorrect Order of Variables

Severity: Minor

Status: Fixed

Code Location: sources/spot_dex.move, L377.

Descriptions: The function `emit_liquidity_added_event()` is called with parameters passed in the wrong order when adding liquidity. It should first be `token_x_amount` and then `token_y_amount`.

Suggestion: Adjust the parameter order.

Resolution: The client has followed our suggestion and fixed the issue.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed**: The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an

endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

MOVEBIT

Securing the Move Ecosystem

https://twitter.com/movebit_

contact@movebit.xyz